1. Physical Set-up

Our hardware components include two personal computers and a SDR (HackRF One). One computer acts as a victim computer (PC1), and the other one acts as the attacker computer (PC2). The victim computer is a Legion Y7000P desktop with 16GB 2666MHz DDR4 SDRAM memory module installed. The receiver has a SDR HackRF One plugged in to capture the Electromagnetic signal, it has a range of 1MHz to 6GHz. We used GNU Radio Companion to process the signals. The covert channel was designed in a normal office environment which will have much background noise, for example from other applications running on the victim computer who will be also accessing memories, or the noise can come from other computers’ memory accessing behaviors, or from other wireless communications. We performed the experiments in the ECE laboratory under one to two meters distance between PC1 and PC2.

1. Victim and Receiver Implementation

For the implementation of PC1, we placed our sender application by downloading the desktop application we wrote for repeated performing reading or not reading the main memory with some rate. Then we run the application in order to manipulate memory access and modulate the Electromagnetic signal generated by DRAM. For the implementation of PC2, we are using HackRF One which is a SDR to receive the signals. To process the Electromagnetic signals, we need a software which we decided to use GNU Radio Companion since it’s one of the most commonly used applications with a modular view. Because GNURadio can’t work on Windows systems, we decided to use a Virtual machine VMware to create a Linux system by Ubuntu. Then we installed GNURadio and a lot of related libraries we need for running GNURadio and HackRF One.

1. Code Running

To control the what’s been sent by the sender, we just needed to edit the main function in “sender.c”. Within the parentheses while(1) { }, simply add sendMessage(1) or sendMessage(0) to send a 1 or a 0. It’s now designed to have a 500ms delay between each bit and the sender will keep repeating the whole string once it’s been turned on.

In order to directly receive the raw signal without any noise from collecting using the SDR. We can run the “read.py” to create a package of binary bits to act directly as the input signals. The string starts from two series of [1,0,1,1,0,1,0,0], followed with the actual bits of the signal and ends with [1,1,1,1,1,1,1,1]. The information combined can be edit in the file “send.txt”.